Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica
Primeiro Semestre de 2022
Prof. Cesar Tadeu Pozzer
Data: 03 / 05 / 2022

Trabalho 2 - Transformada Discreta de Cosseno
OPCIONAL

Ferramentas:

Linguagem C++, utilizando a API Canvas2D (disponível no site da disciplina) e IDE Code::Blocks, compilando com MinGW (disponível na versão 17.12 da IDE Code::Blocks). A leitura do arquivo pode fazer uso das funções da linguagem C. **Não podem ser utilizadas bibliotecas auxiliares. Não pode ser usada a API OpenGL.**

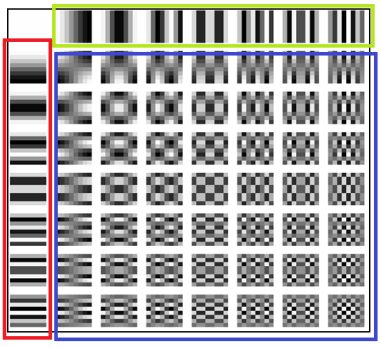
Objetivos:

Explorar quantização, imagens, compressão, transformada cosseno, análise de complexidade.

Descrição:

Desenvolva um programa que aplique a Transformada Discreta de Cosseno (DCT) e sua inversa (IDCT) em uma imagem BMP. O programa deve exibir a imagem original, a imagem reconstruída e a diferença entre as duas imagens. A imagem reconstruída deve ser "idêntica" a original se não aplicado nenhum processo de quantização.

O objetivo do programa é explorar o papel das frequências na reconstrução da imagem. Vamos considerar 3 tipos de frequência: vertical, horizontal e combinação.



Após aplicar a DCT, geram-se os coeficientes de frequência. Desenvolva interfaces gráficas para reduzir ou aplicar frequências específicas da imagem (ou grupos de frequências) e reconstruir novamente, para ver qual efeito isso causa na imagem. A redução de altas frequências causa um borramento. O **aumento** de altas frequências vai causar um realce de bordas**?**

O programa pode operar sobre blocos de 8x8 pixels, 16x16, 32x32, ... ou sobre toda imagem de uma única vez. Para simplificação, vamos considerar apenas o uso de imagens quadradas. Avalie o desempenho da aplicação para diferentes configurações.

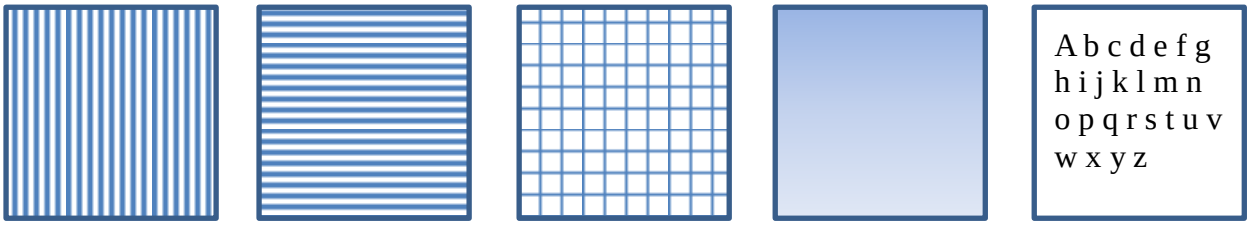
Assumindo N como a largura a da imagem, o algoritmo DCT tem complexidade $O(N^4)$, ou seja, processar 64 blocos de 8x8 é muito mais leve que processar 4 blocos de 32x32, ou um bloco de 64x64, mas ambas imagens têm 64x64 pixels.

- 8^4 = 4.096 * 64 = 262.144
- 32^4 = 1.048.576 * 4 = 4.194.304
- 64^4 = 16.777.216 * 1 = 16.777.216

Procure crie uma janela na canvas2D com 1366 x 768 (resolução padrão de notebooks baratos), mas não maior que 1600 x 1000.

O programa deve estar estruturado em classes. Ex: deve ter uma classe parametrizável para desenhar gráficos. Deve ter uma instância para cada tipo de gráfico (input, idct, dct, diff, etc).

Procure utilizar vários tipos de imagens, como no material de aula: com baixas frequências (fotografias), com altas frequências, imagem com padrão de linhas verticais, linhas horizontais, etc. Pode-se também gerar as imagens de forma procedimental (senos, cossenos, gradientes, etc).



Extras (para nota acima de 9,0):

- Geração de imagens de forma procedimental
- Seja criativo.

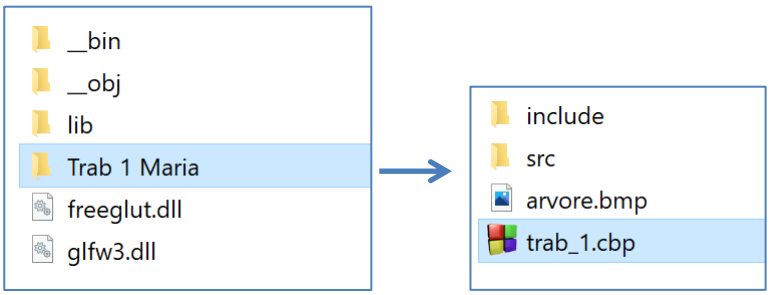
Entrega:

Formato de entrega:

- O arquivo deve ter o nome do aluno (ou uma abreviação nome+sobrenome, ou seja, algo que identifique bem o aluno). O arquivo deve ser enviado pelo Google Classroom, até a data limite.

- O programa deve ser enviado em um arquivo compactado fulano.RAR (fulano = login do aluno. No exemplo abaixo, “Trab 1 Maria”). Dentro deste arquivo deve haver um diretório com o mesmo nome do arquivo e, dentro deste diretório, os arquivos do trabalho (como ilustrado nas seguintes figuras).

- **Deve-se enviar** pastas include, src, projeto code::blocks e imagens
- **Não devem ser enviadas** libs, executáveis, obj, DLLs.



- Antes do envio, certifique-se de que o projeto contido no seu .rar funciona em qualquer diretório que ele seja colocado e não dependa de outros arquivos não incluídos no envio do trabalho.
- Retire todo código não utilizado no trabalho (arquivos, métodos, variáveis, etc), bem como printiefis() de depuração.

Antes do envio, teste se o projeto contido no seu .rar funciona em qualquer diretório que ele seja colocado.

Critérios de avaliação:

- Documentação: descrever no cabeçalho de cada arquivo a ideia geral do código e comentar o que cada método e classe faz.
- Clean code: estrutura do código e nomeação de métodos, classes e variáveis devem ser fáceis de ler e entender. Procurar manter o código o mais simples e organizado possível.
- README: incluir um arquivo “README.txt” contendo informações sobre quais funcionalidades foram implementadas (requisitos e extras) e instruções de uso do programa caso o aluno julgue necessário ou caso tenha sido implementado uma funcionalidade extra que exija explicação.
- Pontualidade: Trabalhos não entregues na data não serão avaliados e receberão nota zero.
- Funcionalidade: o programa deve satisfazer todos os requisitos. Programas que não compilarem ou que não atenderem nenhum requisito receberão nota 0 (zero).